Research Journal of Applied Sciences, Engineering and Technology 6(10): 1711-1719, 2013
DOI:10.19026/rjaset.6.3893
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: September 25, 2012 Accepted: November 23, 2012 Published: July 20, 2013

Research Article

A Methodology for Computational Efficiency Improvement of Z-Matrix in Power System Fault Analysis Using Evolutionary Algorithms

[1]Sajjad Abedi, [2]Arash Alimardani, [2]Mehrdad Abedi and [2]Seyed Hossein Hosseinian
[1]Islamic Azad University, Damavand Branch, Damavand, Iran
[2]Power System Analysis Lab., Electrical Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., 15875-4413, Tehran, Iran

Abstract: This study presents a novel and comparative approach to select an optimal path during direct Z-Matrix building process using Evolutionary Algorithms (EAs). The proposed evolutionary methods are based on selection of series and shunt elements (i.e., lines, transformers and generators) in optimal order for minimizing computation time. The proposed evolutionary methods are tested on IEEE 14-bus benchmark and the results are compared. The proposed method is also arranged to find the optimal path considering all possible network alterations due to all possible faults to avoid the requirement for repeating the calculation process for each single line fault. The comparative results indicate the feasibility and effectiveness of the method to find the optimal path in Z-matrix building process and considerably diminishing the time consumption for Z-matrix modification.

Keywords: Computation time, evolutionary algorithms, IEEE 14-bus benchmark, impedance matrix, optimal path

INTRODUCTION

Bus impedance matrix (Z-matrix) plays an important role in the field of power system analysis and operation. However, the process of Z-matrix building is complicated and more time consuming compared to bus admittance matrix formulation (Peterson *et al*., 1989). Nevertheless, the information contained in Z-matrix is more pronounced (Peterson and Makram, 1989; Grainger and Stevenson, 1994; Bergen and Vittal, 2000; Tianmin and Baozhu, 2009). For instance, the diagonal element related to specific bus in Z-matrix presents thevenin impedance at this bus.

Z-matrix has many effective and useful applications in the field of fault analysis, voltage sag calculations, contingency study, economic dispatch and relay setting (Guochen, 1995). In addition, Z-matrix reflects a linear relationship between injected current and bus voltages for distribution network analysis (Shipley *et al*., 1966; Reitan and Kruempel, 1969; Reitan, 1980; Makram *et al*., 1989; Glover and Sarma, 1994; Wei *et al*., 2005).

Z-matrix building process is most commonly based on direct and indirect methods. Indirect method requires Y-matrix inversion. Y-matrix formation is relatively simple, however, adding and removing an element due to system modification would require repetitive matrix inversion.

In order to find new and modified version of Z-matrix, this event leads to complicated and time consuming process especially in a large electric power system. Direct method which is appeared in most electrical text books can be applied for Z-matrix building using a step-by-step procedure, while in each step one element can be added accordingly. The important advantage of this method is that, after Z-matrix building termination any changes or modification can be implemented to existing Z-matrix by simple algorithms to produce a new Z-matrix (Grainger and Stevenson, 1994; Miller and Goldberg, 1995.

Yue *et al*. (2004) presented a decomposition method based on spanning tree and two decomposition matrices. However, this method is not able to cope with mutual inductances and is not well suited for strongly meshed power networks.

In spite of the importance of the Z-matrix, its usefulness has been primarily constrained by the computational burden due to building process. Direct method for Z-matrix building is based on a step by step procedure which in each step, different choices regarding different element to be added are on the table. Thus, different paths related to available choices provide different computation time in Z-matrix building process. Therefore, among all available paths including set of elements to be added, only one path can be nominated as optimal path with minimum computation time. The most recent report based on detecting the optimal path in Z-matrix building process is appeared in (Ranjbar *et al*., 2008), which employs Genetic

Corresponding Author: Sajjad Abedi, Power System Analysis Lab., Electrical Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., 15875-4413, Tehran, Iran

Algorithm (GA) for optimization purposes. However, there is no comparison with other evolutionary algorithms for result validation. Furthermore, the network alterations due to short circuit in the lines of the network and its influence on the calculation time are not taken into consideration.

In the present study, an effort has been made to present a comparative and novel approach for detecting the optimal path during Z-matrix building process for efficient short circuit computation in large power systems. The aim is to develop an optimization problem to find the best element order in the direct Z-Matrix building method and minimize the calculation time. Considering all possible network alterations due to fault occurrence in all network lines, it is noteworthy to state that based on the proposed method, for only one time, only one unique overall optimal path should be determined which is applicable for all of the mentioned network alterations and any further short-circuit computation, resulting in considerable improvement in computational efficiency. In order to nominate a solution method to the optimization problem with best convergence quality and speed, four popular evolutionary optimization algorithms are employed and compared, including: conventional Particle Swarm Optimization (PSO) as well as its three modified versions, Genetic Algorithms (GAs), Shuffled Frog Leaping Algorithm (SFLA) and Differential Evolution (DE) algorithm and its modified version.

PROBLEM FORMULATION

Basic concepts: The Z-matrix describes the relationship between bus voltages vector and injected current (Ranjbar *et al*., 2008), as follows:

$$[V_{Bus}] = [Z_{Matrix}][I_{Bus}] \tag{1}$$

The diagonal elements of the Z-matrix, called driving-point impedance or Thevenin impedance are defined as:

$$Z_{ii} = \frac{V_i}{I_i}\bigg|_{\substack{I_j = 0 \\ j \neq i}} \tag{2}$$

The off-diagonal elements named the transfer impedance are defined as:

$$Z_{ji} = \frac{V_j}{I_i}\bigg|_{\substack{I_j = 0 \\ j \neq i}} \tag{3}$$

Using these definitions the conventional direct method of Z-matrix building has been developed (Grainger and Stevenson, 1994; Tianmin and Baozhu, 2009). This method deals with elements one by one and categorizes them into 4 types:

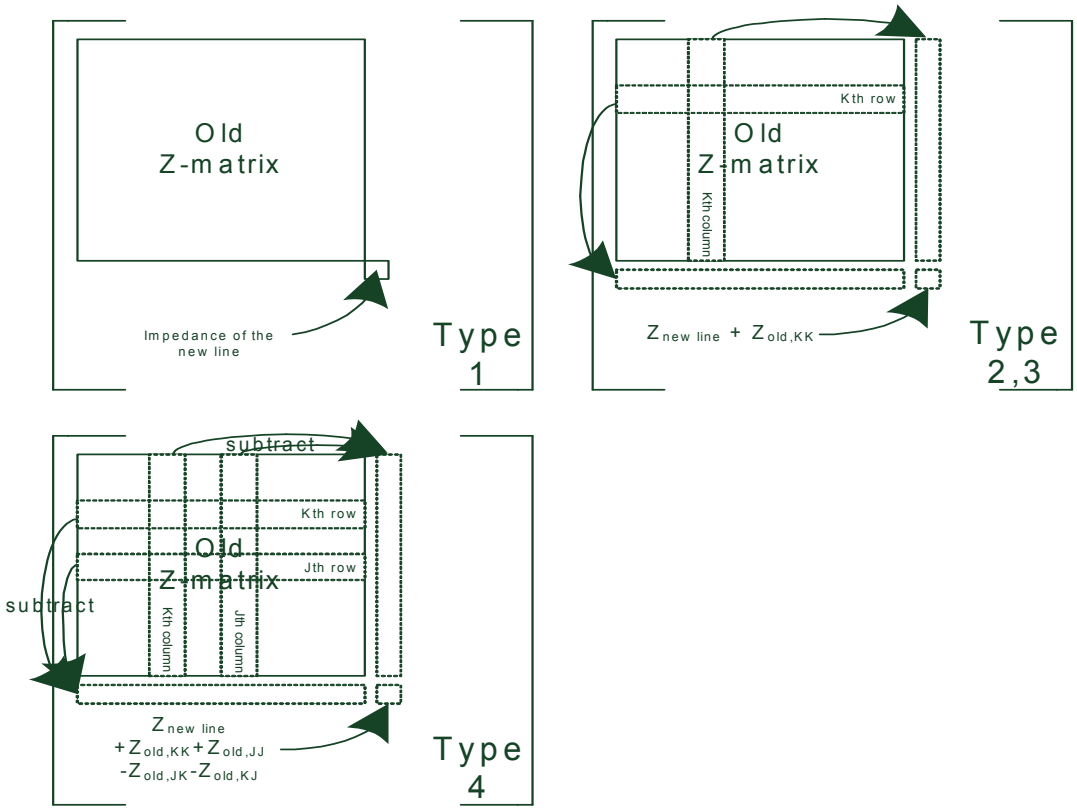

Fig. 1: The procedure of line injection

Type 1: Adding an element from a new bus to the reference bus.
Type 2: Adding an element from a new bus to an existing bus.
Type 3: Adding an element from an existing bus to the reference bus.
Type 4: Adding an element between two existing buses.

Fitness function: In order to evaluate the computation time of each possible order of elements, a fitness function dependent on the type of the current element should be defined. As mentioned, there are four types of elements to be added in Z-matrix building process. Figure 1 illustrates the mechanism of each type of element injection into Z-matrix. Type 3 and type 4 require a Kron reduction afterwards. Equation (4) to (7) represent the calculation time of each type of element respectively:

$$T_1 = t_R \quad (4)$$

$$T_2 = 2nt_R + t_A + t_R \quad (5)$$

$$T_3 = 2nt_R + n^2\left(t_M + t_D + t_S + t_R\right) + t_A + t_R \quad (6)$$

$$T_4 = 2n\left(t_R + t_S\right) + n^2\left(t_M + t_D + t_S + t_R\right) + 2t_A + 2t_S + t_R \quad (7)$$

In these equations, n is the dimension of the matrix when an element is assembled, t_R, t_S, t_A, t_D and t_M are the required time for replacement, subtraction, addition, division and multiplication operators, respectively. Each element is connected between two buses. Depending on the type of element to be added (type 1 to 4) one of the Eq. (4) to (7) is taken into consideration, correspondingly. So the fitness function for each candidate solution can be defined as follows:

$$Ztime = 0 \quad (8)$$

For each element in order of the candidate solution, if the bus is type *i*:

$$Ztime = Ztime + T_i \quad (9)$$

The value of t_R, t_S, t_A, t_D and t_M depend on the architecture of each computer and the software in use. In this study, to determine these operational times, a simple method is implemented. A matrix with the same dimension as Z-bus matrix with random complex variables is defined.

Then, each operation is performed on every element of this matrix. This procedure is repeated for 100000 iterations for each operation and the total calculation time is obtained. Finally, the average time of each operation per element is determined and also normalized.

EVOLUTIONARY ALGORITHMS

In this following, Genetic Algorithm, Particle Swarm Optimization algorithm, Shuffled Frog Leaping algorithm and Differential Evolution algorithm and some of their modified versions are briefly presented.

Genetic algorithm: Genetic Algorithm (GA) is a particular class of evolutionary algorithms. This search technique is used in finding exact or approximate solutions to optimization problems. Every solution is represented in the form of a string called chromosome which consists of a set of variables called genes. Each chromosome is evaluated by the objective function (Elbeltagi *et al*., 2005).

The GA used in this study, begins with a set of chromosomes randomly generated within the feasible space. In each iteration, GA tries to improve the chromosome with worst fitness by generating an offspring through crossover or mutation procedures. Crossover is a natural process between parent chromosomes and is given a rate ranging from 0.6 to 1 (Elbeltagi *et al*., 2005). Figure 2 illustrates the crossover operation between two randomly chosen parents. The offspring may be affected by mutation. In this case, some genes change by chance. This makes the chromosomes spread out the search space and may avoid being trapped in local minimum. Mutation rate is usually assumed less than 0.1 (Goldberg, 1989).

Particle swarm optimization: Particle Swarm Optimization (PSO) is a multi-agent search technique, which is inspired by the social behavior of a flock of birds searching for food (Kennedy and Eberhart, 1995). Each bird in the flock is called a particle and the flock is referred to as a swarm. Each particle travels through multidimensional search space looking for the best position (global optimum), by adjusting its position according to its own experience as well as the experience of its adjacent particles (Elbeltagi *et al*., 2005).

In this notation, X and V are particle coordinates and its corresponding velocity in the search space, respectively. The best position of a particle and the best position of all particles are recorded and represented as P_{best} and G_{best}, respectively. In each iteration, the velocity and position of particles are calculated as follows:

$$V_i(t+1) = wV_i(t) + c_1 \times Rand() \times (P_{best,i} - X_i(t)) + c_2 \times Rand() \times (G_{best,i} - X_i(t)) \quad (10)$$

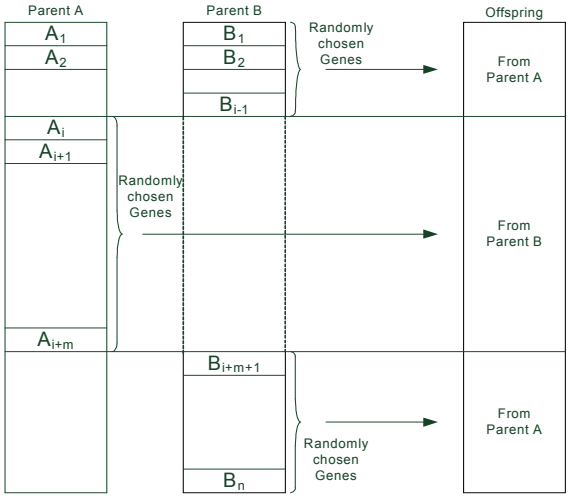

Fig. 2: Genetic algorithm offspring generating process

Table 1: Parameters of PSO versions

PSO parameters	C-PSO	CI-PSO	LDI-PSO	T-1 PSO
Inertia weight (w)	1	0.729844	0.9-0.4	0.7290
Learning factor c_1	2	2.01	2	1.4944
Learning factor c_2	2	2.01	2	1.4944

$$X_i(t+1) = X_i(t) + V_i(t) \tag{11}$$

where,
c_1 & c_2 : Learning factors
w : Called the inertia weight

The iteration process continues until the termination condition is satisfied. Conventional PSO (C-PSO), Inertia Constant PSO (IC-PSO), Linearly Decreasing Inertia PSO (LDI-PSO) and Type 1 PSO (T-1 PSO) (Shi and Eberhart, 1998; Valle *et al*., 2008) are applied in this study. Table 1 indicates the parameters of these algorithms.

Shuffled frog leaping algorithm: The SFL algorithm originally developed as a population-based met heuristic to perform an informed heuristic search using mathematical functions to find a solution of a combinatorial optimization problem (Amiri *et al*., 2009). It combines the benefits of both the genetic-based Memetic Algorithm (MA) and the social behavior-based particle swarm optimization algorithm.

In SFL algorithm, there is a population of possible solutions defined by a set of frogs that is divided into subgroups called memeplexes, each performing a local search. After a defined number of memetic evolution steps, ideas are passed among memeplexes in a shuffling process. The local search and the shuffling process continue until defined convergence criteria are satisfied (Elbeltagi *et al*., 2005).

At first, an initial population of *P* frogs is created randomly within the feasible space. For an *S* variable problem, the i^{th} frog is represented as $X_i = (x_{i1}, x_{i2},\ldots, x_{iS})$. Then, the frogs are sorted in a descending order according to their fitness. Then, the whole Population (P) is separated into m memeplexes, each containing n frogs. In this procedure, the first frog moves to the first memeplex, the second frog moves to the second memeplex, frog m moves to the m^{th} memeplex and frog m+1 goes back to the first memeplex, etc. Within each memeplex, position of frogs with the best and worst fitnesses is determined as X_b and X_w, respectively. Also, position of frog with the global best fitness is determined as X_g. Then, in each memeplex, a process is applied to improve only the frog with the worst fitness (not all frogs) in each cycle as follows:

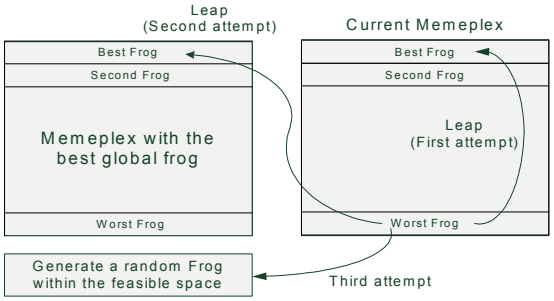

Fig. 3: Shuffled frog leaping algorithm improvement attempts

$$D_i = Rand() \times (X_b - X_w) \tag{12}$$

$$X_{w_{NEW}} = X_w + D_i \tag{13}$$

where, Rand() is a random number between 0 and 1. This operation is shown in Fig. 3. As the first attempt, if this process generates a better solution, the worst frog will be replaced. Otherwise, the calculations in (12) and (13) are repeated with replacement of X_b by X_g (second attempt). If no improvement becomes possible in this case, then a new solution is randomly generated within the feasible space to replace the worst frog (third attempt). Then, the calculations continue for a specific number of iterations (Elbeltagi *et al*., 2005).

After a pre-specified number of memetic evolutionary steps within each memeplex, to ensure global exploration, ideas passed within memeplexes are combined in the shuffling process (Amiri *et al*., 2009). The local search and the shuffling process continue until convergence criteria are satisfied. Figure 3 shows the main idea of this algorithm.

Differential evolution algorithm: Differential evolution algorithm, introduced by Price *et al*. (2005), is a simple population based, stochastic evolutionary algorithm for global optimization and is capable of handling non-differentiable, nonlinear and multi-modal objective functions (Varadarajan and Swarup, 2008). In DEA, the population consists of real-valued vectors with dimension *D* that equals the number of design

parameters. The size of the population is adjusted by the parameter N_p (Price *et al*., 2005). The initial population is uniformly distributed in the search space. Each variable *k* in an individual *i* in the generation *G* is initialized within its boundaries $x_{k,min}$ and $x_{k,max}$. At each generation, two operators, namely mutation and crossover (recombination), are applied to each individual; thus producing the new population. Then, a selection phase takes place, where each individual of the new population is compared to the corresponding individual of the old population and the best of them is selected as a member of the population in the next generation. In the following, the evolutionary operators are briefly described. The first step is the mutation that can be described as follows:

$$V_{i,G+1} = X_{r1,G} + F.(X_{r2,G} - X_{r3,G}) \quad (14)$$

where,

$X_{ri,G}$: Randomly chosen vector among the population in the generation G
F : A constant within (0, 2)
$V_{i,G+1}$: The trial vector

If $X_{r1,G}$ is replaced by $X_{best,G}$, another form of the presented DE (R-DE) called B-DE will be formed (Abedi *et al*., 2012). In the second step called the crossover step, Eq. (15) is used to determine the trial vector that may replace the current vector in the next population with the probability of Crossover constant (CR) which is between 0 and 1 (Price and Storn, 1997):

$$U_{j,i,G+1} = \begin{cases} V_{j,i,G+1} & if\ rand_{j,i} \le CR\ or\ j = i \\ X_{j,i,G+1} & otherwise \end{cases} \quad (15)$$

where, $rand_{j,I}$ is a random number generated within the range 0 and 1. Finally, the selection phase is performed and the generated vector is tested by being compared with the best vector of prior iteration. These steps are repeated in times of a defined number of iterations or the algorithm is terminated if the stop circumstances are confirmed.

OPTIMAL PATH FINDING METHODOLOGY USING EVOLUTIONARY ALGORITHMS

In order to implement each EA method, a vector containing the element order of a specified network is defined and a set of such vectors is developed to form a population; this is a common concept in all the algorithms used. When the population is generated, the aforementioned fitness function should be calculated for every vector considering the following constraints.

There are two uncommon constraints to be considered in the optimization process using any of the mentioned evolutionary algorithms. First, every vector of elements order has to start with a generator to establish the connection to the reference bus.

The second constraint, which is very hard to satisfy in case of employing evolutionary algorithms, is that every element to be added in each step of the Z-matrix building process should have a direct or indirect path to the reference bus through the previously added elements. In other words, at least one of the buses of the new element should be one of the buses that are already added. Hence, there are three significant defects in employing any EA for this optimization problem.

Firstly, generally when an evolutionary method generates a new population, the values in every vector of the population are not integers. This problem may be overcome by rounding the variables.

Secondly, even if the variables are integers, it is possible that some variables have the same value. It is obvious that no two variables in each vector must be equal with each other.

Thirdly, even if the variables are integers and mutually different, it is very improbable that the vector satisfies both the constraints together.

To overcome these difficulties, a novel method in vector generation in each population is proposed. This method can be applied in any evolutionary algorithms that are used for the optimization problem.

In this problem, the second constraint is severe on the solution vector which determines a specified relation between each element with the previous one. In other words, since in building Z-matrix, an element could be added only under one of the four underlying conditions, the operators in evolutionary algorithms (crossover, …) should generate new vectors, which satisfies this constraint on each element. However, generating a high-dimensional vector from existing ones which satisfies this constraint on each element of a vector is very improbable. The main reason that in this study, each employed optimization method is briefly described is to clarify this issue. This might take a great deal of trails until a vector with satisfying property (mentioned constraint) on all elements is generated.

Therefore, in this method, instead of generating vectors of all elements altogether and checking the constraints to be satisfied, the vector is discretized into 1×1 arrays each containing only one element number and every step of the optimization algorithm is performed on this scalar. For example, in GA, crossover operator is performed on a single gene. Then the constraints are checked for each single scalar. This means that it should be checked that the resulted value by the operator, which is the number of a bus, is a bus connected to the previously added to the Z-matrix or is one of the generators. In this case, if the generated scalar does not satisfy the constraints, the performed steps is repeated only for the same scalar not for the total array. If after for example 100 times the steps repeated and the constraints are still unsatisfied, only the current scalar is randomly generated. After generating this element (gene), the next one will be produced. This procedure is illustrated in the flowchart shown in Fig. 4.

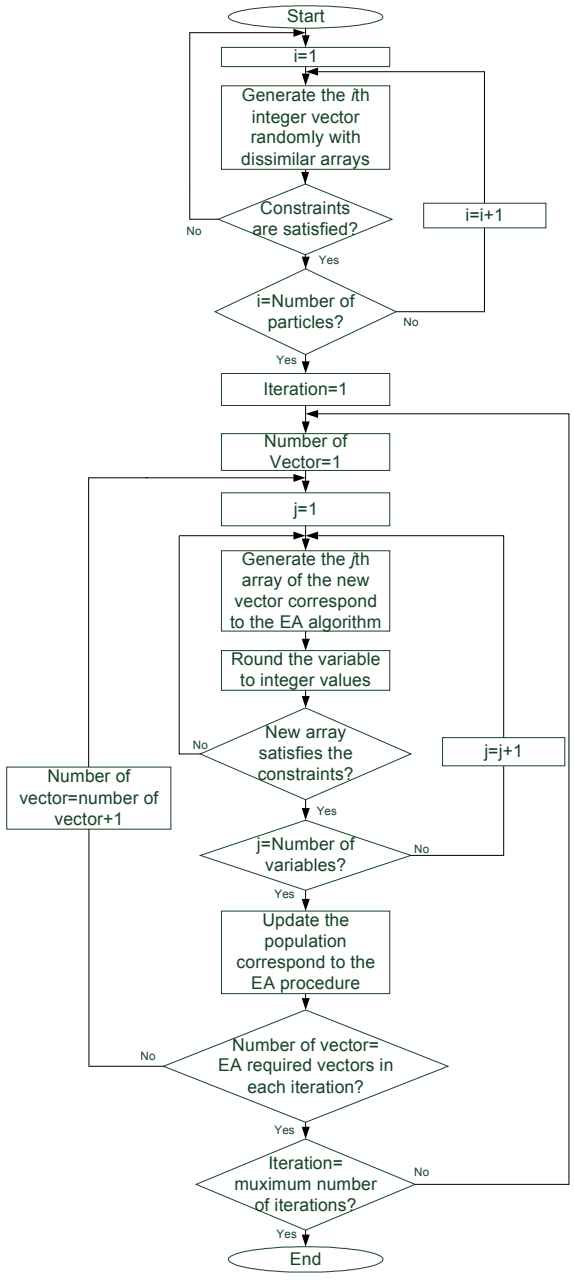

Fig. 4: Proposed method to handle the constraints

Case study-single network: The proposed Z-matrix building method was tested on IEEE 14-bus benchmark (Christie, 1999a). As the algorithm requires the buses numbered, the number of individuals is given in Table 2. The parameters t_R, t_S, t_A, t_D and t_M are calculated on a E-7200 Pentium with a 2 GBs RAM on MATLAB 7.6. Table 3 depicts the per unit values of each operation time. The chosen base value is t_R = 4.42e-8 sec.

Figure 5 and 6 demonstrate the iterative convergence of best run for each EA method. With regard to the randomness of the heuristic algorithms, many trials with different initializations should be experienced to prove if the algorithm is robust. The comparison of performance of employed algorithms

Table 2: Numbering of IEEE 14-bus power system

Element number	From bus	To bus
1	0	1
2	0	8
3	0	2
4	0	3
5	0	6
6	1	5
7	1	2
8	2	3
9	2	5
10	2	4
11	3	4
12	4	7
13	4	9
14	4	5
15	5	6
16	6	11
17	6	12
18	6	13
19	7	8
20	7	9
21	9	14
22	9	10
23	10	11
24	12	13
25	13	14

Table 3: Calculation time of each operator (P.U.)

Replacement	Addition	Subtraction	Multiply	Division
1	0.8042	0.7702	7.9876	8.1566

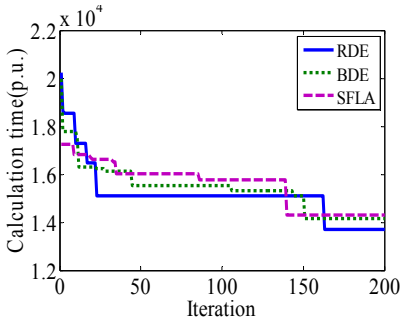

Fig. 5: Iterative convergence of employed EAs

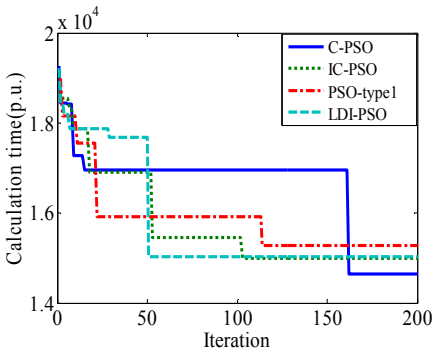

Fig. 6: Iterative convergence of employed EAs

Table 4: Statistical results of different EAs through 50 trials

Compared algorithm	Best Z-matrix time (p.u.)	Mean Z-matrix time (p.u.)	Worst Z-matrix time (p.u.)
C-PSO	14643	15930	16705
CI-PSO	15002	16080	16710
LDI-PSO	15280	16021	16876
T-1 PSO	15038	16249	17112
GA	14731	16313	17270
B-DE	14157	14950	15572
R-DE	13704	15039	15691
SFLA	14296	15566	16328

Table 5: Convergence time comparison among applied evolutionary algorithms

Compared algorithm	Shortest convergence time	Average convergence time	Longest convergence time
C-PSO	58.69	266.15	1181.90
IC-PSO	78.89	209.11	362.26
LDI-PSO	58.50	229.06	467.85
T-1 PSO	73.74	451.13	1131.50
GA	57.68	125.91	1956.00
R-DE	66.74	90.41	113.69
B-DE	67.81	92.94	111.98
SFLA	33.45	697.31	1213.80

after 50 independent runs is shown in Table 4 and 5. Moreover, the average computation time of randomly generated paths for this network is computed as 32258 p.u. which is distinctly comparable with the determined optimal path calculation time shown in Table 4.

The result of implementation of each optimization algorithm on the benchmark to find the optimal path is presented in Table 6. Considering fitness and convergence time of different algorithms, Table 4 to 6 depict that R-DE is the most efficient method for this application. GA and SFLA are the next efficient algorithms. Figure 7 demonstrates the IEEE 14-bus diagram and the global optimal path of Z-matrix building resulted by DE algorithm.

Case study-short-circuited networks: In some applications of Z-Matrix such as fault analysis and voltage sag calculation after a fault, the network topology is altered. To determine a unique optimal path, applicable in all possible network topologies, the objective function should be modified such that the average calculation time of all the networks is minimized. In other words, we are looking for one optimal path with no need to further determination of the optimal path as various faults occur.

As a case study, the unique optimal path is determined for IEEE 14-bus power system. As DE had the best performance among the applied algorithms, only this method is employed in this part of the case study. The result of this optimization is presented in Table 7. Moreover, the average computation time of 1000 randomly generated paths are calculated for comparison. This is done by sequentially selecting the elements randomly among the feasible solutions. For this network, the result is 29683 p.u., which is distinctly comparable with the determined optimal path calculation time shown in Table 7.

Table 6: Comparison of optimization results in the IEEE 14-bus power system

Modification process	C-PSO	CI-PSO	LDI-PSO	GA	Type 1 PSO	B-DE	R-DE	SFLA
1	1	1	2	3	4	3	1	1
2	6	6	4	10	1	5	6	6
3	3	7	3	9	6	7	9	7
4	7	3	8	6	8	6	4	3
5	14	9	1	7	3	1	10	8
6	10	10	6	4	10	9	7	9
7	4	14	14	8	9	15	8	10
8	8	8	9	14	11	8	11	14
9	11	4	11	1	14	4	5	4
10	9	11	10	11	2	11	14	11
11	12	12	7	13	12	10	15	13
12	13	13	19	12	7	14	17	12
13	15	20	13	20	5	13	18	2
14	5	5	20	19	19	20	16	19
15	20	15	12	2	15	12	24	20
16	2	18	5	15	13	18	13	15
17	17	25	15	5	20	19	3	5
18	19	21	18	17	18	21	12	18
19	24	17	25	21	25	24	22	17
20	18	24	21	18	21	25	23	24
21	21	22	24	25	24	16	25	25
22	25	16	17	24	17	23	20	21
23	16	2	22	16	22	22	21	22
24	22	23	16	22	16	2	2	23
25	23	19	23	23	23	17	19	16
Convergence time (p.u.)	341.39	261.63	305.57	93.85	207.16	87.99	88.71	871.6
Building process time (p.u.)	14643	15002	15280	14731	15038	14157	13704	14296

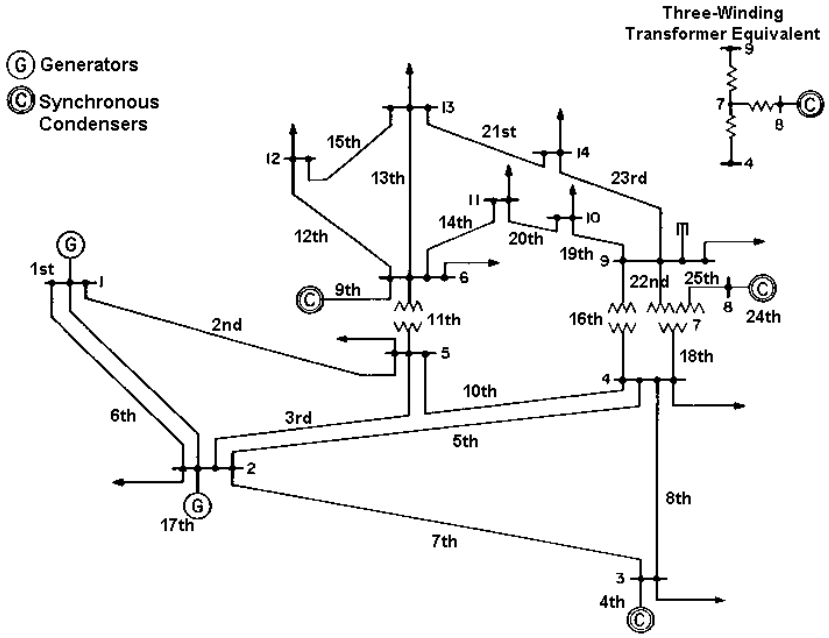

Fig. 7: IEEE 14-bus network and line tags showing the order of the global optimal path

Table 7: Optimization results for short-circuited networks case study

Modification process	Line number
1	4
2	3
3	11
4	1
5	6
6	9
7	14
8	7
9	12
10	8
11	19
12	10
13	13
14	2
15	20
16	15
17	21
18	18
19	16
20	25
21	5
22	22
23	17
24	24
25	23
Convergence time (p.u.)	16.13
Building process time (p.u.)	14920

As another instance, a simulation on a bigger network such as IEEE 118-bus benchmark (Christie, 1999b) is performed. If the aim is to calculate sag voltages caused by all short circuits each time occurred in each line of the network, then 186 different possibilities for fault occurrence exist with respect to the network toplogy in Z-matix calculation (The network contains 186 lines). Taking the advantage of the proposed method, for only one time, only one optimal path should be calculated which is applicable for all of these cases. Therefore, the time taken for computation of the optimal path by implementing the proposed method is not of great concern especially in large networks.

Similar to the 14-bus case, the unique optimal path in the 118-bus system is obtained by minimizing the average time of Z-matix computation of all the mentioned topology alterations. To give an insight to the effectiveness of the applied method, the average Z-matrix computation time of 1000 randomly generated paths in the 118-bus system is calculated as 301228 p.u. and compared to the case of applying the calculated optimal path. The result indicates that we gain 51.2% saving time.

CONCLUSION

In this study, four different algorithms including PSO and three of its modified versions, i.e., GA, SFLA, DE and one of its modified versions are implemented and compared to find the best element order in the direct Z-Matrix building method and minimize the calculation time. To demonstrate the performance of the proposed method, it has been applied to IEEE 14-bus test system. Based on the test results, the best optimization method to determine the optimal path is discussed. Furthermore, as the most significant application of Z-Matrix is in fault analysis and calculation of voltage sags and because during a fault occurrence in any of test network elements, the network topology changes, the proposed method is arranged to find the optimal path considering all possible network

alterations to avoid the requirement for repeating the calculation process for each single line fault. Taking the advantage of the proposed method, for only one time, only one optimal path should be calculated which is applicable for all of the mentioned network alterations and any further short-circuit computation. This is better demonstrated with the 118 bus case study. Results demonstrate the effectiveness and capability of this method to considerably reduce the time consumption for Z-matrix modification which is useful especially in large-scale power systems.

REFERENCES

Abedi, S., A. Alimardani, G. Gharehpetian, G. Riahy and S. Hosseinian, 2012. A comprehensive method for optimal power management and design of hybrid RES-based autonomous energy systems. Renew. Sust. Energ. Rev., 16(3): 1577-1587.

Amiri, B., M. Fathian and A. Maroosi, 2009. Application of shuffled frog-leaping algorithm on clustering. Int. J. Adv. Manuf. Technol., 45: 199-209.

Bergen, R. and V. Vittal, 2000. Power Systems Analysis. 2nd Edn., Prentice-Hall Inc., NJ, USA, pp: 619.

Christie, 1999a. The IEEE 14-Bus Test System (Online). Retrieved from: http: // www. ee. Washington .edu/ research/ pstca/ pf14/pg_ tca14bus.htm

Christie, 1999b. The IEEE 118-Bus Test System (Online). Retrieved from: http:// www. ee. washington. edu/research /pstca/pf118/ pg_tca118 bus. htm.

Elbeltagi, E., T. Hegazy and D. Grierson, 2005. Comparison among five evolutionary-based algorithms. Adv. Eng. Info., 19(1): 43-53.

Glover, J.D. and M.S. Sarma, 1994. Power System Analysis and Design. Cengage Learning, United States.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Reading Mass, Addison-Wesley Publishing Co., NY.

Grainger, J.J. and W.D. Stevenson, 1994. Power System Analysis. McGraw-Hill, New York.

Guochen, C., 1995. Fast calculation of power system with variation structure at any complex fault. Proc. Chinese Soc. Electr. Eng., 5(15): 354-360.

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimization. Proceeding of the IEEE International Conference on Neural Networks (ICNN), 4: 1942-1948.

Makram, E.B., K.P. Thornton and H.E. Brown, 1989. Selection of lines to be switched to eliminate overloaded lines using a Z-matrix method. IEEE T. Power Syst., 4(2): 653-657.

Miller, B.L. and D.E. Goldberg, 1995. Genetic algorithms, tournament selection and the effects of noise. Comp. Syst., 9: 193-212.

Peterson, W.L. and E.B. Makram, 1989. A Z-matrix building algorithm for unbalanced power systems with mutually coupled lines. Proceeding of the 21st Southeastern Symposium on System Theory. Tallahassee, FL, pp: 9-12.

Peterson, W.L., E.B. Makram and T.L. Bakdwin, 1989. A generalized PC based bus impedance matrix building algorithm. Proceeding of the IEEE Energy and Information Technologies in the Southeast, 2: 432-436.

Price, K. and R. Storn, 1997. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optimiz., 11: 341-359.

Price, K.V., R. Storn and J.A. Lampinen, 2005. Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series, Springer, Berlin, Germany.

Ranjbar, A.H., H. Omranpour, M. Abedi and G.B. Gharehpetian, 2008. A novel approach for a Z-matrix building process using genetic algorithm. Proceeding of the IEEE 2nd International Power and Energy Conference, Johor Bahru, pp: 1161-1165.

Reitan, K.D., 1980. A new method using the bus-impedance matrix model for short-circuits calculations. Proc. IEEE, 68(8): 1027-1030.

Reitan, K.D. and K.C. Kruempel, 1969. Modification of the bus impedance matrix for system changes involving mutual couplings. Proc. IEEE, 57(8): 1432-1433.

Shi, Y. and R. Eberhart, 1998. A modified particle swarm optimizer. Proceeding of the IEEE World Congress on Computational Intelligence, pp: 69-73.

Shipley, R.B., N. Sato, D.W. Coleman and C.F. Watts, 1966. Direct calculation of power system stability using the impedance matrix. IEEE T. Power Ap. Syst., 85(7): 777-782.

Tianmin, F. and L. Baozhu, 2009. A novel algorithm for Building Z-matrix of Electric Power Network including CCVS. Proceeding of the Asia-Pacific Power and Energy Engineering Conference (APPEEC), Wuhan.

Valle, Y.D., G.K. Venayagamoorthy, S. Mohagheghi, J. Hernandez and R.G. Harley, 2008. Particle swarm optimization: Basic concepts, variants and applications in power systems. IEEE T. Evolut. Comput., 12(2): 171-195.

Varadarajan, M. and K.S. Swarup, 2008. Differential evolutionary algorithm for optimal reactive power dispatch. Int. J. Elect. Power Energy Syst., 30: 435-441.

Wei, L., B. Hai, F. Jiyue and X. Xiao, 2005. Problem of loss allocation based on power components theory. Proc. Chinese Soc. Electr. Eng., 25: 157-160.

Yue, Q., W. Yu and F. Lu, 2004. A novel algorithm for building Z-Matrix. Proceeding of the IEEE PES Power Systems Conference and Exposition, 1: 124-129.